Guide to using Contexts in XKS Fingerprints
Version 1.0

Example 1

```
    $a = cc('pk') and (web_search('jihad') or document_body('planning
    for jihad'))
```

Definition

(S//REL) Contextual expressions are those that restrict the search space for a particular expression. In example 1 above, we are looking for the string '`jihad`' only in the normalized text of a web search, and '`planning for jihad`' only in the context of the UTF-8 normalized text of an office document.

GENESIS provides a number of different *context types* depending on the function of the context:

- *hash* – exact match
- *scan* – perform a keyword/regex scan on contextual text
- *latlong* (geobox) – perform an R-tree geobox lookup on the latlong location
- *prefix* – matches the longest prefix of the context and tasked term
- *appid/fingerprint/topic* – triggers based on appids, fingerprints, and topic logic
- *extracted_file* – allows hooking on raw extracted files transmitted on the network

Technical Note

(U//FOUO) The difference between a "hash" and a "scan" context is that "scan" means that a full keyword scan will be executed against that context's data which means the keyword will still hit if it's a substring of a larger word (think of it as being wildcarded on both ends). A hash lookup must be an exact match – which is much faster and less taxing on front-end resources.

(S//REL) For example, web_search is a *scan* so web_search('jihad') will hit on web searches like:

"I want to participate in jihad"
"How do I avoid jihad"
"jihadi"
"bigjihad"

What is Contextual Logic?

(S//REL) Contextual logic is the ability to look for keywords, regular expressions, geo-boxes, and other events purely within a specified a scope (context). While this may not sound like a big deal, currently the *only* context that current DNI processing sensors provide is that of "strong-selector" where an email address/chat handle/ip address is extracted from a known application type and looked up against a list of known targets.

(S//REL) Contextual logic allows the creation of advanced analytics with extreme precision. For example, if as an analyst needs to find all people in Kabul performing web searches on Jihad, how could this be done? For starters, tasking the term Jihad in CADENCE or other similar systems would result in voluminous collect as the keyword hit on every news web page, blog entry, RSS feed etc. The result would be a ton of data that would ultimately be useless to the analyst and waste precious exfil bandwidth.

(S//REL) So who could I ask the question "Show me all the people doing web searches on Jihad from Kabul?" Well for starters, the system would need to understand web searches. What protocol is used for relaying the text of a search from your browser to the search site's data centers? The system must not only be capable of identifying this traffic, but also of processing it and extracting out the text of the web search. Well it turns out that most search engines uses the HTTP protocol (just like every other web site out there) and the search terms are url-encoded and are passed as the url argument "q". XKEYSCORE (or other equivalent system) will extract the "q" url argument and then normalize the text by url-decoding it. After normalization, the text is passed to the GENESIS context for scanning against all search terms NSA analysts are looking for.

(S//REL) So this is how we would task the web search:

```
fingerprint('web_search/extremist/jihad') =
        web_search('jihad' or 'mojahadeen') ;
```

(S//REL) But wait – we're not done yet. The analyst requested that the expression only be true if the person was physically located in Kabul.

(S//SI//REL) To execute the geographic this part of the question, the GENESIS engine performs an NKB IPGEO lookup against all sessions. The country and city codes are then passed to the contexts relating to country and city. The city code tasking for Kabul will fire.

This is the updated fingerprint:

```
fingerprint('web_search/extremist/jihad') =
        web_search('jihad' or 'mojahadeen') and city('Kabul');
```

(S//SI//REL) So now, we have the web-search context for Jihad firing, and the city tasking for Kabul firing. Both of these events are then combined in the GENESIS engine's Boolean evaluator where the "AND" is evaluated and the resulting fingerprint fired, tagging the session as being potentially interesting.

(U//FOUO) This is a very simple example, but very powerful. The concept of contexts gives analysts power that was never possible with CADENCE. It allows the tasking of combinations of soft terms in context that together form a very strong event.

What is the syntax for Contextual Logic

(U//FOUO) The syntax for contexts is:

```
    context_name(boolean_expression)
```

(U//FOUO) Contexts themselves are Boolean expressions that allow composition of expressions involving the use of many contexts. The example in the previous section demonstrated this use, which is very natural:

```
fingerprint('web_search/extremist/jihad') =
        web_search('jihad' or 'mojahadeen') and city('Kabul');
```

Technical Note
(S//REL) You may not compose expressions that include one context inside of another. For example the expression web_search('jihad' or city('Kabul')) is somewhat meaningless and not permitted.

Dynamic Contexts

(U//FOUO) Dynamic contexts in GENESIS are those that can be expressed by regular expressions. For example, if you wanted to write expressions that operated on the HTML title of a web page only and GENESIS didn't have a pre-defined context to serve this purpose (it does), then you could write the following expression:

```
my_html_title = scan {
  /<title>(.*?)</title>/;
};

fingerprint('web_site/bad_guy') =
        my_html_title('bomb' and 'making');
```

(U//FOUO) The first expression defines a context named "`my_html_title`". The type of the context is "`scan`" – i.e. perform a keyword scan of tasked terms against the extracted text.

(U//FOUO) The fingerprint makes use of the dynamic context by looking for the keywords '`bomb`' and '`making`' within the context of the extracted HTML title.

(S//REL) Here is another example – this time a dynamic expression for Yahoo chat:

```
yahoo_chat_notify = hash {
  /49\xc0\x80typing\xc0\x801\xc0\x80(.{3,40})\xc0\x80/c;
  /49\xc0\x80contactinfo\xc80\x801\x80(.{3,40})\xc0\x80/c;
};
```

```
fingerprint('badguy') =
       yahoo_chat_notify('badguy');
```

(U//FOUO) In this case, we have given two regular expressions that will perform extraction against the data stream. The type of the context is “hash” meaning that the extracted text will be looked up in a hash table – in that hash table will be any tasking applied to that context, in this case “badguy” has been used in that context below.

Technical Note
(S//REL) Notice that in the yahoo_chat_notify example above the “c” following the regular expressions. It is a requirement that all expressions should have the same case sensitivity setting. The same requirement applies for contexts of type “hash”. The case sensitivity applied not only to the regular expression running against the raw traffic but also the actual lookup/scan of the extracted text against terms tasked against that context.

Advanced Contexts

Context presence

(S//REL) Sometimes the presence of a context in traffic is all we need. For example, if the GENESIS is to be used to filter traffic being forwarded to a site-store, we may want to pass all chat sessions from Mumbai to Pakistan that have content. Anyone who has looked at chat traffic very quickly realizes that there are volumes of presence messages sent, much of which are not that interesting, and then there is the chat that has content. How could we find all traffic with actual chat content? Well we have a “chat_body” context, but how do I task “not null” in reference to a particular context? Well a hack would be to task “chat_body(not ‘junk’)”, then every chat session that does not contain the word “junk” will be selected, however that is not a very elegant solution, and in fact is very inefficient. GENESIS allows tasking of a not null simply by providing an empty context as follows:

```
fingerprint('mumbai/chat') =
       cc('pk') and city('mumbai') and chat_body();
```

Appendix A – Context Catalog

File Transfers

filename
Description:

(U//FOUO) Every utf-8 normalized filename seen in traffic is passed to this context. Examples are (but not limited to) files transmitted in SMTP, POP3, IMAP, HTTP Responses, HTTP Posts, FTP sessions, and MIME-encoded header. Note that the filename refers to just the filename and not the entire path+filename.

Aliases:
fname

Context Type:
Full Scan

Eample:
filename(/wimax.{0,30}setup/)
filename('passport.jpg')

file_ext

Description:
(U//FOUO) Every utf-8 normalized file extension seen in traffic is passed to this context. Examples (but not limited to) are files transmitted in SMTP, POP3, IMAP, HTTP Responses, HTTP Posts, FTP sessions, and MIME-encoded header.

Aliases:
ext

Context Type:
Hash

Eample:
file_ext('jpg' or 'jpeg')

path

Description:
(U//FOUO) Every utf-8 normalized file path seen in traffic is passed to this context. Examples are (but not limited to) files transmitted in SMTP, POP3, IMAP, HTTP Responses, HTTP Posts, FTP sessions, and MIME-encoded header.

Aliases:
dir

Context Type:
Full Scan

Eample:
path(/Document and Settings\\[a-z]{20}/ or '/home/test')

HTTP Activity

web_search

Description:
(U//FOUO) The normalized extracted text from web searches. The system extracts search terms from Google, Microsoft, News sites and search term leakage from the Referer line of HTTP headers. In addition it will extract and use spelling correction from the HTML server response – so if the target mis-spells but the search engine corrects the expression will still evaluate true.

Aliases:
search

Context Type:
Full Scan

Eample:
web_search('ricin' or 'plague')

html_title

Description:
(U//FOUO) The normalized extracted text web page titles.

Aliases:
http_title

Context Type:
Full Scan

Eample:
html_title('how to' and 'bomb')

http_url

Description:
(U//FOUO) Every URL from HTTP GET and POST commands.

Aliases:
url

Context Type:
Full Scan

Eample:
http_url('/mail/inbox?action=delete')

http_url_args

Description:
(U//FOUO) All arguments given as part of a URL (ie. all text following the '?' in a URL string)

Aliases:
url

Context Type:
Full Scan

Eample:
http_url('action=delete')

http_host

Description:
(U//FOUO) The "Host:" name given in the http header.

Aliases:

<none>

Context Type:
Full Scan

Eample:
http_host('yahoo.com')

http_server

Description:
(U//FOUO) The "Server:" type name in the http header.

Aliases:
<none>

Context Type:
Full Scan

Eample:
http_host('GWS/2.1' or 'Apache')

http_referer

Description:
(U//FOUO) The "Referer:" URL given in the HTTP header.

Aliases:
<none>

Context Type:
Full Scan

Eample:
http_referer('http://badwebsite/cp?action=show')

http_language

Description:
(U//FOUO) The normalized two letter iso-6393 language code as inferred from any http and or html header info.

Aliases:
<none>

Context Type:
Full Scan

Eample:
http_language('fa' or 'de')
http_language(not 'en')

http_cookie

Description:
(U//FOUO) The "Cookie:" field given in the http header.

Aliases:
<none>

Context Type:
Full Scan

Eample:
http_cookie(/PREF=\d\d[a-z]/)

http_user_agent

Description:
(U//FOUO) The "User-Agent:" field given in the http header. This is the name of the browser the client is using (eg Firefox, or Internet Explorer).

Aliases:
user_agent
http_browser
browser

Context Type:
Full Scan

Eample:
http_user_agent(/Mozilla\/[45]/ or 'Chrome')

GEO Info

map_latlong

Description:
(U//FOUO) A geo box around the location of a map view zoom on services such as Google Earth, Google Maps, Microsoft Live Earth etc.

Aliases:
map, map_zoom, zoom

Context Type:
latlong box

Eample:
map_latlong('1, 2, 3, 4')

to_cc

Description:
(S//REL) The destination country based on IP address - IPGEO lookup.

Aliases:
ip_to_cc

Context Type:
Hash

Eample:
to_cc('ir' or 'pk')

from_cc

Description:
(S//REL) The source country based on IP address - IPGEO lookup.

Aliases:
ip_from_cc

Context Type:
Hash

Eample:
from_cc('ir' or 'pk')

cc

Description:
(S//REL) The country (either to OR from) based on IP address - IPGEO lookup.

Aliases:
ip_cc

Context Type:
Hash
Aliased to from_cc and to_cc.

Eample:
cc('ir' or 'pk')

to_latlong

Description:
(S//REL) A geo box around the destination latitude and longitude based on IP Address – IPGEO lookup.
NOTE – this context is currently disabled for performance

Aliases:
<none>

Context Type:
latlong box

Eample:
to_latlong('1, 2, 3, 4')

from_latlong

Description:
(S//REL) A geo box around the source latitude and longitude based on IP Address – IPGEO lookup.
NOTE – this context is currently disabled for performance

Aliases:
<none>

Context Type:
latlong box

Eample:
to_latlong('1, 2, 3, 4')

latlong

Description:
(S//REL) A geo box around the source or destination latitude and longitude based on IP Address –IPGEO lookup.
NOTE – this context is currently disabled for performance

Aliases:
<none>

Context Type:
latlong box
Aliased to from_latlong and to_latlong.

Eample:
latlong('1, 2, 3, 4')

to_city

Description:
(S//REL) The destination city based on IP address - IPGEO IPGEO lookup.

Aliases:
to_town

Context Type:
Hash

Eample:
to_city('Islamabad')

from_city

Description:
(S//REL) The source city based on IP address - IPGEO lookup.

Aliases:
from_town

Context Type:
Hash

Eample:
from_city('Islamabad')

city

Description:
(S//REL) The source or destination city based on IP address - IPGEO lookup.

Aliases:
from_town

Context Type:
Hash
Aliased to from_city and to_city.

Eample:
from_city('Islamabad')

SRI Metadata

sigad

Description:
(S//REL) The site designator (SIGAD).

Aliases:
<none>

Context Type:
Scan

Eample:
sigad('US-1234')

casenotation

Description:
(S//REL) The collection site casenotation (signal designator).

Aliases:
case_notation

Context Type:
Scan

Eample:
casenotation('ABC1234')

block_num

Description:
(S//REL) The session block number. If a TCP/IP session exceeds a maximum size (typically 10Mbytes) the session is fragmented and assigned a one-up non-zero block number. If a session is block zero it is the one and only fragment.

Aliases:
block

Context Type:
Scan

Eample:
block_num('1' or '2')

sigint_header

Description:
(S//REL) The tasked expression will only hit on terms in a USSID-124 SIGINT header, and not in the body of the communications.

Aliases:
header

Context Type:
Scan

Eample:

sigint_header('ILCC' and 'FM IP')

sigint_body

Description:
(S//REL) The tasked expression will only hit on terms in the communications content and not in USSID-124 or protocol headers.

Aliases:
body

Context Type:
Scan

Eample:
sigint_header('ILCC' and 'FM IP')

sigint_body

Description:
(S//REL) The tasked expression will only hit on terms in the communications content and not in USSID-124 or protocol headers.

Aliases:
body

Context Type:
Scan

Eample:
sigint_header('ILCC' and 'FM IP')

Targetting/Strong Selectors/Soft Selectors

realm

Description:
(S//REL) The selector and its corresponding realm. The syntax for this is in the standard form of "badguy<yahoo>" where "badguy" is the username and "<yahoo>" is the realm. realm() tasking is

Aliases:
<none>

Context Type:
Hash

Eample:
realm('badguy<yahoo>")

strong_selector

Description:
(S//REL) Automatically attempts to determine realm (selector could be email address, cookie or other) and then creates DECODEORDAIN-style permutations to task in the main scanner engine for the given target. Given the number of permutations for a single

target using this syntax can have a significant performance impact on the system if too many targets are tasked this way.
NOTE: realm() tasking should be used in preference to strong_selector().

Aliases:
<none>

Context Type:
Hash

Eample:
strong_selector('badguy@yahoo.com") // an email address
strong_selector('ABCDEF") // a cookie value

email_address

Description:
(S//REL) Automatically attempts to determine realm for the given email address and then creates DECODEORDAIN-style permutations to task in the main scanner engine for the given target. Given the number of permutations for a single target using this syntax can have a significant performance impact on the system if too many targets are tasked this way.
NOTE: realm() tasking should be used in preference to email_address().

Aliases:
<none>

Context Type:
Hash

Eample:
email_address('badguy@yahoo.com")

raw_email

Description:
(S//REL) Tasks the given email address with no permutations in the raw_email context. The raw_email context is fed email addresses from the XKEYSCORE email address extractor which scans all traffic looking for "@" and then uses heuristics to determine if the "@" is part of an email address. Note that the output from STARPROC is also scanned for email addresses, as is the fully UTF-8 normalized application-layer processed content.

Aliases:
<none>

Context Type:
Hash

Eample:
email_address('badguy@yahoo.com")

email_cc

Description:
(S//REL) The country code associated with an email address. For example if the email address is badguy@yahoo.co.de the country code is 'de'.

Aliases:

<none>

Context Type:
Hash

Eample:
email_cc('ir' or 'sa')

category

Description:
(S//REL) The CADENCE category that has evaluated true on this session. Note that the CADENCE category must have a valid FIST entry for this context to fire.

Aliases:
cat

Context Type:
Hash

Eample:
category('2344')

category_priority

Description:
(S//REL) The priority of the CADENCE category that generated a hit. Note that the CADENCE category must have a valid FIST entry for this context to fire.

Aliases:
cat_pri

Context Type:
Hash

Eample:
category_priority(1)

tnd

Description:
(S//REL) Looks up all phone numbers found in signature blocks and other content as well as phone numbers found in VoIP.

The TND field will accept traditional 'X' and 'Y' wildcards. 'X' will match any number and 'Y' is permissible at the beginning and will match any number of digits at the beginning of the number.

Aliases:
pstn
phone
mobile
cell
phone_number
msisdn

Context Type:
tnd_lookup

Eample:
tnd('5551234')

imei

Description:
(S//REL) All IMEIs found in network protocols or HTTP headers (as seen in many cellular providers).

Aliases:
<none>

Context Type:
tnd_lookup

Eample:
imei('123456789012345)

imsi

Description:
(S//REL) All IMSIs found in network protocols or HTTP headers (as seen in many cellular providers).

Aliases:
<none>

Context Type:
tnd_lookup

Eample:
imsi('123456789012345)

Office Documents

document_title

Description:
(U//FOUO) The title of the office document.

(U//FOUO) Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_title

Context Type:
Scan

Eample:
document_title('situation report')

document_subject

Description:
(U//FOUO) The subject of the office document.

(U//FOUO) Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_subject

Context Type:
Scan

Eample:
document_subject('latest figures')

document_author

Description:
(U//FOUO) The author of the office document. Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_author

Context Type:
Scan

Eample:
document_author('badguy')

document_org

Description:
(U//FOUO) The authoring organization of the office document.

(U//FOUO) Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_org

Context Type:
Scan

Eample:
document_org('PTCL'c)

document_hash

Description:
(U//FOUO) The MD5 sum the office document/images from within the document.

(U//FOUO) Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_hash

Context Type:
Hash

Eample:
document_hash('cfd2c677a42bd0919dd8e37f7ac9bcf7')

document_language

Description:
(U//FOUO) The language of the Office document. The language can be determined from the document properties and or statistcal analysis of the underlying text. All languages are normalized to two letter iso-6393 language codes.

(U//FOUO) Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_language

Context Type:
Hash

Eample:
document_language('de')

document_body

Description:
(U//FOUO) The UTF-8 normalized text of the Office document.

(U//FOUO) Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_body

Context Type:
Scan

Eample:
document_body('how to' and 'build' and ('bomb' or 'weapon'))

Communications Content

document_body

Description:
(U//FOUO) The UTF-8 normalized text of the Office document.

(U//FOUO) Office documents include (but are not limited to) Microsoft Office, Open Office, Google Docs and Spreadsheets.

Aliases:
doc_body

Context Type:
Scan

Eample:
document_body('how to' and 'build' and ('bomb' or 'weapon'))

document_email_body

Description:
(U//FOUO) The UTF-8 normalized text of all office document and email bodies.

Aliases:
doc_email_body

Context Type:
Scan
Aliased to email_body and document_body.

Eample:
document_email_body('how to' and 'build' and ('bomb' or 'weapon'))

communication_body

Description:
(U//FOUO) The UTF-8 normalized text of all office document, email, and chat bodies.

Aliases:
doc_email_body

Context Type:
Scan
Aliased to email_body, chat_body and document_body.

Eample:
communication_body('how to' and 'build' and ('bomb' or 'weapon'))

email_body

Description:
(U//FOUO) The UTF-8 normalized text of all email bodies.

Aliases:
email_body

Context Type:
Scan

Eample:
email_body('how to' and 'build' and ('bomb' or 'weapon'))

chat_body

Description:
(U//FOUO) The UTF-8 normalized text of all chat bodies.

Aliases:
chat_body

Context Type:
Scan

Eample:
chat_body('how to' and 'build' and ('bomb' or 'weapon'))

calendar_body

Description:
(U//FOUO) The UTF-8 normalized text of all calendars. An example is Google Calander.

Aliases:
chat_body

Context Type:
Scan

Eample:
calendar_body('wedding')

archive_files

Description:
(U//FOUO) Matches a list of files from within an archive. For example is a ZIP file is transmitted, all names of files within are passed to this context.

Aliases:
archive_body
compressed_filenames

Context Type:
Scan

Eample:
archive_files('bad.dll' or 'virus.doc')

http_post_body

Description:

(U//FOUO) The UTF-8 normalized text HTTP url-encoded POSTs.

Aliases:
<none>

Context Type:
Scan

Eample:
http_post_body('action=send' and 'badguy@yahoo')

language

Description:
(U//FOUO) The iso-6393 language code for either documents or web activity.

Aliases:
<none>

Context Type:
Hash
Aliased to doc_language and http_language.

Eample:
language('ar')

title

Description:
(U//FOUO) The title of either office documents or HTML pages.

Aliases:
<none>

Context Type:
Scan
Aliased to doc_title and html_title.

Eample:
title('bomb making')

Protocol Metadata

protocol

Description:
(U//FOUO) The textual form of the IP next protocol.

Aliases:
<none>

Context Type:
internal

Eample:
protocol('TCP')
protocol('UDP')
protocol('ICMP')

ip_next_protocol

Description:
(U//FOUO) The textual form of the IP next protocol.

Aliases:
<none>

Context Type:
internal

Eample:
ip_next_protocol(17)

from_ip

Description:
(U//FOUO) The source IP address of the session.

Aliases:
fromip

Context Type:
Scan

Eample:
from_ip('127.0.0.1')

to_ip

Description:
(U//FOUO) The destination IP address of the session.

Aliases:
toip

Context Type:
Scan

Eample:
to_ip('127.0.0.1')

ip

Description:
(U//FOUO) The source or destination IP address of the session.

Aliases:
<none>

Context Type:
Scan

Eample:
ip('127.0.0.1')

from_port

Description:
(U//FOUO) The source TCP or UDP port number.

Aliases:
fromport

Context Type:
internal

Eample:
from_port(22)

to_port

Description:
(U//FOUO) The destination TCP or UDP port number.

Aliases:
toport

Context Type:
internal

Eample:
toport(22)

port

Description:
(U//FOUO) The source or destination TCP or UDP port number.

Aliases:
<none>

Context Type:
internal

Eample:
port(22)

ip_subnet

Description:
(U//FOUO) IP subnet in CIDR notation.

Aliases:
<none>

Context Type:
Scan

Eample:
ip_subnet('7.211.143.148/24')

mac_address

Description:
(U//FOUO) The MAC address of the target network device.

Aliases:
mac

Context Type:
Scan

Eample:
mac_address('00:16:3E:3F:BD:EF')

Checkpoints

checkpoint

Description:
(S//REL) A system defined checkpoint to ensure guarenteed order-of-execution. This is typically used when a follow-on-action must be performed after some predefined processing step, for example access to pre-parsed HTTP header information is only available after the internal XKEYSCORE http_parser plugin has executed.

Aliases:
<none>

Context Type:
Hash

Eample:
checkpoint('http_parser')

Misc

appid

Description:
(U//FOUO) The application ID of the session.
NOTE: to prevent infinite recursion only one level of indirection is permitted when including appid() as part of another boolean expression.

Aliases:
<none>

Context Type:
internal

Eample:
appid(/mail.*/)

preappid

Description:
(U//FOUO) The pre-application ID of the session. A pre-application ID is a boolean expression for an application that fired on a session, but did not necessarily win (based on priority). For example a Yahoo webmail session will probably first be identified as HTTP, then Yahoo, and then finally we see a string indicating that the traffic is mail. As the decision was being made each of the intermediate appids can generate a pre-appid event.

NOTE: to prevent infinite recursion only one level of indirection is permitted when including preappid() as part of another boolean expression.

Aliases:
<none>

Context Type:
internal

Eample:
preappid(/mail.*/)

fingerprint

Description:
(U//FOUO) A fingerprint that fired on a session.

NOTE: to prevent infinite recursion only one level of indirection is permitted when including fingerprint() as part of another boolean expression.

Aliases:
<none>

Context Type:
internal

Eample:
fingerprint(/encryption\/office.*/)

topic

Description:
(U//FOUO) A topic that fired on a session based of a previous topic definition.

NOTE: to prevent infinite recursion only one level of indirection is permitted when including topic() as part of another boolean expression.

Aliases:
<none>

Context Type:
internal

Eample:
topic('wmd/pakistan' and /wmd\/bio.*/)